REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 19 March 1998	3. REPORT TYPE AND DATES COVERED Final Report 17 August 1997 - 17 March 1998

4. TITLE AND SUBTITLE Development of a Commercial FOM for 3D Interactive Entertainment Applications

5. FUNDING NUMBERS

C M67004-97-C-0049

6. AUTHORS Andreas Kemkes, Douglas Rogers and Richard Vestewig

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Perceptronics, Inc.
21010 Erwin Street
Woodland Hills, CA 91367

8. PERFORMING ORGANIZATION REPORT NUMBER
PWH98-1

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Simulation, Training and Instrumentation Command (STRICOM)
Attn: AMSTI-ET (Mark McAuliffe)
12350 Research Parkway
Orlando, FL 32826-3276

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES N/A

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public releases; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Report developed under SBIR contract. This report describes the development of a High Level Architecture (HLA) federates and Federation Object Model (FOM) to show the viability of this approach for multiparticipant, networked games and other entertainment applications. A simple tennis-like game called Tong, representative of a particular application class, was developed and represented using HLA development requirements. The game was played on a testbed consisting of two PC platforms connected by ethernet, with communication accomplished by the HLA Run Time Infrastructure (RTI) 1.0-2 for Windows. Both standard 2D and a unique autostereo (no glasses or headtracking) 3D display, which shows 3D images with parallax , were used for visualization. Design and implementation was performed for three key issues for interactive games: transfer of ownership, viewpoint sharing and multiple levels of detail viewing, including device-independent abstractions. The report concludes that HLA principles and RTI services are appropriate for commercial applications including networked games, Location Based Entertainment, medical and education.

14. SUBJECT TERMS SBIR Report, HLA, FOM, Distributed Interactive Simulation, 3D displays, real-time commercial interactive games, federate/federation, view sharing

15. NUMBER OF PAGES 26

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE SAR	19. SECURITY CLASSIFICATION OF ABSTRACT SAR	20. LIMITATION OF ABSTRACT UL

TABLE OF CONTENTS

LIST OF FIGURES

Development of a Commercial FOM for 3D Interactive Entertainment Applications

Small Business Innovation Research (SBIR) Phase I
Topic Number OSD97-003
Contract Number M67004-97-C-0049
Final Report (0001AF)

Submitted by:
Perceptronics, Inc.
21010 Erwin Street
Woodland Hills, CA 91367

SECTION I: INTRODUCTION AND OVERVIEW

1.1 Introduction

This is the Final Report for the SBIR contract Development of a Commercial FOM for 3D Interactive Entertainment Applications, which was developed in response to Topic OSD97-003, HLA Commercial Applications in Simulation. The goal of this project is to develop a High Level Architecture (HLA) Federation Object Model (FOM) which will meet requirements for the realtime, interactive, networked games and simulations which are becoming more numerous and sophisticated. A second goal of this project is to meet specific requirements for true 3D displays with parallax, to show feasibility of integrating this type of display into a game environment, and also to affirm the hardware independence of HLA. The commercial target for this technology is on-line or Location Based Entertainment gaming; and additional military or medical applications where networking of advanced displays will increase capability beyond that provided by standard 2D displays.

To accomplish these goals, we used the HLA Run Time Infrastructure (RTI) in a testbed containing both the Litton/Infinity autostereo 3D display and conventional displays. The RTI provided communication for a specially developed, two-player interactive game which allowed realtime play across both a LAN and WAN. Using HLA development principles, we developed a Federation Object Model (FOM) for the game.

Technical challenges for use of the RTI and FOM development principles for networking of 3D displays are transfer of ownership of objects, sharing viewpoints between players and observers, and multiple levels of detail viewing. We addressed these both by analysis and federation design, and by testing critical elements in the testbed. We demonstrated the applicability of using the RTI for interactive games in the testbed. The testbed and the prototype game application were shown at IITSEC 97 to STRICOM representatives and other attendees.

The team for this SBIR includes Perceptronics, Inc. (prime contractor); and subcontractors Litton Guidance and Control Systems, Northridge, CA; and Infinity Multimedia International, Sherman Oaks, CA.

1.2 Summary of Results

Perceptronics and its team members were able to meet the major objectives of this project, including both analysis and development of key HLA modules, and demonstration of applicability of the HLA RTI for communication in a realtime interactive game. We demonstrated the feasibility of using HLA simulation development principles and methods to construct an interactive game FOM, send realtime messages over the RTI to support game play, and develop FOM elements to meet requirements of 3D and 2D displays to demonstrate display independence. In particular:

1. We developed a FOM for a realtime, interactive networked game representative of a more general class of entertainment applications to which HLA principles can be applied.

2. Using HLA principles, we designed and implemented areas important for achieving realtime play of interactive games, including ownership management, sharing of viewpoints, and multiple levels of detail viewing. We used RTI services including declaration management to achieve these implementations.

3. We developed a testbed which included distributed PC-based hosts and player stations with either a standard 2D display or the autostereo 3D display. Communication was achieved using the Windows NT version of the HLA Runtime Infrastructure (RTI). This testbed was used to demonstrate use of HLA principles and the RTI to show

feasibility of HLA application to commercial interactive games and other entertainment applications.

1.3 Conclusions

We were able to demonstrate a playable realization of a networked 3D game based on HLA/RTI principles, and addressing several major implementation issues, including transfer of ownership and sharing viewpoints. Further, we developed concepts in the HLA framework, especially multiple levels of detail viewing, which are relevant for multiplayer online games where very high detail is not necessary for all players, and which may slow down game performance without improving the game experience. Overall, our findings are encouraging for continued use of the RTI in interactive game applications, and for HLA design and development principles. Based on the findings of Phase I, our future work will focus on applying the open architecture implied by HLA to commercial online games.

SECTION II: DETAILED DESCRIPTION OF PHASE I ANALYSES, IMPLEMENTATIONS AND DEMONSTRATIONS

2.1 Phase I Objectives

Our principle objectives for Phase I were to:

1. Develop a Federation Object Model (FOM) using HLA methodology and principles of a prototype networked realtime interactive game (Section 2.2).
2. Develop a testbed for analysis of the RTI, performance, and display independence of the prototype networked interactive game (Section 2.3).
3. Apply HLA ownership management services as a mechanism to reduce network effects, such as latency (Section 2.4).
4. Develop and implement capability of sharing viewpoints among game participants, including required abstractions in order to allow view sharing to be display-independent (Section 2.5).
5. Use HLA data distribution management services to implement multiple levels of detail viewing (Section 2.6).

The Phase I objectives are discussed in the noted sections.

2.2 Tong Game and Federation Object Model (FOM)

A tennis-like 3D game application, named Tong was designed and implemented (see Figure 2-1). The game allowed us to look at several important aspects and issues of high-speed, networked 3D games. The critical issues examined include transfers of ownership, sharing of viewpoints, and viewing at multiple levels of detail. All issues examined have broader relevance for this general domain of applications.

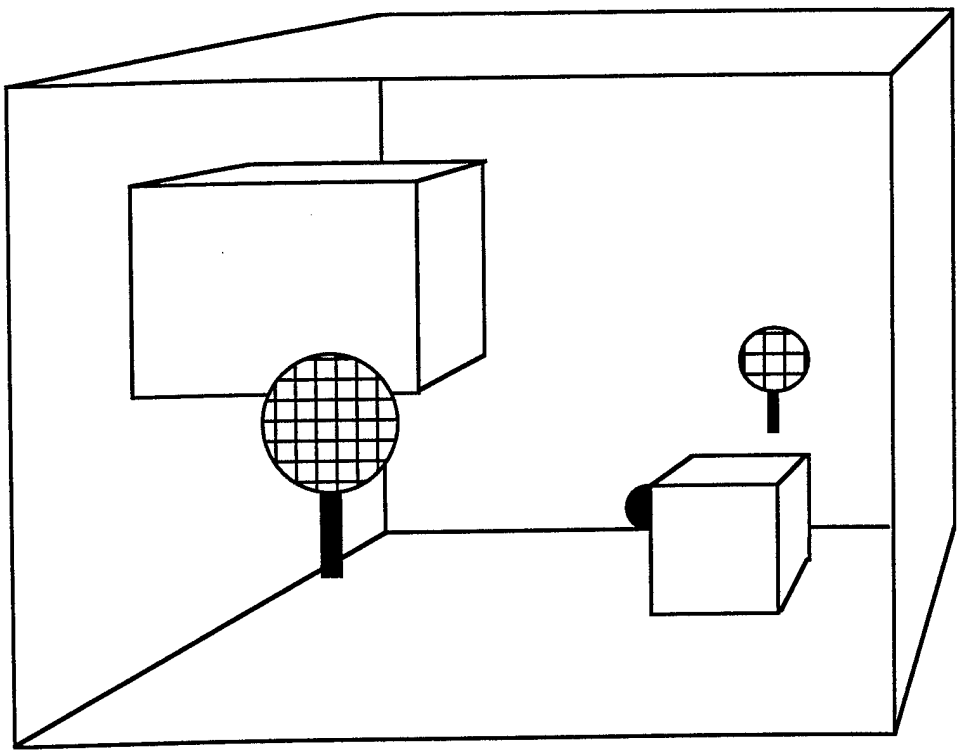

Figure 2-1: The Tong Game

2.2.1 Tong Game Description.

The main participants in the Tong game are players and observers (see Figure 2-2). Players actively determine the outcome of the game, while observers are merely viewing the course of the game. A ball is hit back and forth between the two players. The playing space is a corridor that also contains static and dynamic objects that temporarily obstruct the view. This forces the players to change their viewpoint in order to see the ball. (For the player viewing the game on the autostereo display, viewpoint change consists of moving from side to side to look around objects in the corridor, using the parallax feature of the autostereo display.) The ball may also bounce off the corridor walls or collide with obstacles in the corridor.

The experience of playing and observing Tong is different depending on what display is viewed. On a standard 2D display, the ball moving in the game corridor appears to be larger or smaller depending on whether the ball is moving toward or away from the player, using size as a standard distance cue. There is also the option of including scale measurement lines "attached" to the ball to show the player how far down the corridor the

ball has traveled. On the 3D display, the game is also played in a corridor, but with the autostereo 3D providing depth and distance cues as well as the size cues. On the 3D display, the player can "look around" obstacles by moving his head from side to side to reacquire the ball after it has moved behind an obstacle, a feature made possible by the 3D display's parallax capability; obstacles cannot be "looked around" on the 2D display, but rather the player must wait until the ball reappears from behind the obstacle before he can reacquire it.

The observer can only watch the action, not influence the game play. However, he can view the game from any vantage point within the game environment.

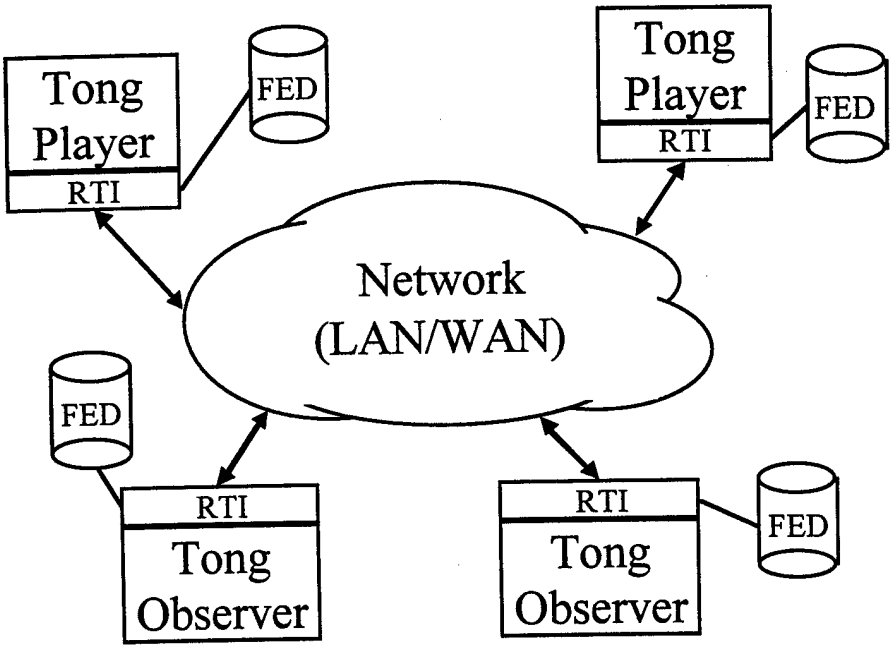

Figure 2-2: Schematic Diagram of Tong

2.2.2 HLA Federation Object Model (FOM) Development

We developed a simple Federation Object Model (FOM) for the prototype networked interactive game using HLA development methodology and principles. The FOM consists of two classes of objects. The first class is TongObject, with attributes Location, Orientation, and Mass. This object described relevant physics of the game. Subclasses of TongObject are Racquet, describing the racquet for each player; and Ball, describing the ball hit back and forth. The second class is Camera, with Location and Orientation

attributes. The Camera class is described in more detail in the Sharing Viewpoints portion of this report.

2.3 Testbed Implementation

2.3.1 Distributed Network Demonstration Platform.

We have established a distributed network demonstration platform to show the applicability of the HLA approach to commercial interactive distributed game applications. The platform consists of Pentium Pro and Pentium II hardware with Windows NT. RTI 1.0-2 has been installed to provide the communication backbone. The distributed network demonstration platform currently supports three sites as shown in Figure 2-3. The platforms were resident at both Perceptronics and Litton.

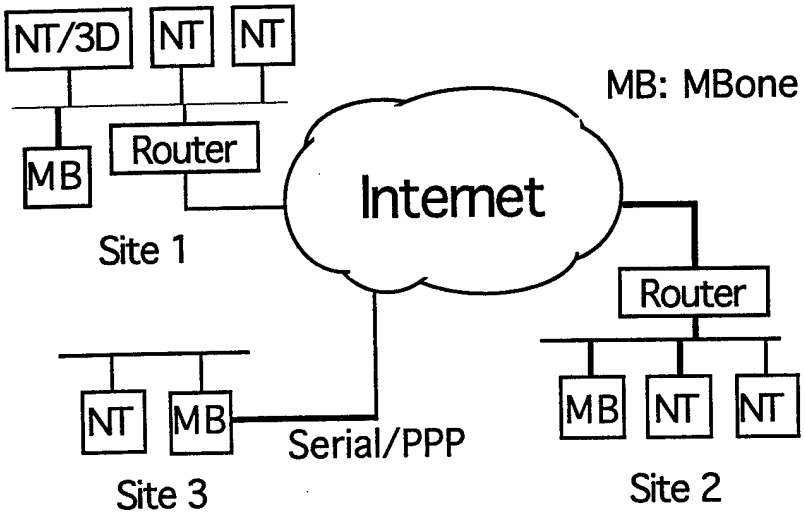

Figure 2-3: Three-Site Testbed Block Diagram

All sites are used for performance and usability tests. Site 1 and site 2 also host the primary development environment, and were replicated for a demonstration at IITSEC 1997 of Tong using RTI 1.0-2. Site 3 represents a typical home-user platform and is connected via a modem/serial line to the Internet. The MBone implementation has been provided in anticipation of supporting the multicast capabilities of RTI 1.3 that can be used

by multicast tunneling through the MBone. Currently, the test sites do not make use of the multicast tunneling capability.

One dedicated NT platform is connected to the 25" autostereo display prototype, which is configured as shown in Figure 2-5, and the other platforms use a standard (2D perspective) 3D display. The platform connected to the autostereo display, in the network with 2D displays, allows us to evaluate display independence of the testbed using RTI 1.0-2

2.3.2 Litton Autostereo 3D Display.

The Litton/Infinity 3D technology overcomes the limitations of conventional stereo three-dimensional displays by using time-multiplexing rather than spatial multiplexing for the stereo effect. The images corresponding to the various viewing angles are flashed on the image generating device sequentially in time, each occupying the full screen. In this configuration the views can also share a common optical train, reducing complexity and making it easier to scale the system for more views. The various views are channeled into the appropriate viewing slice in space by means of a special display backlight arrangement or by use of an electronic shutter scheme. Figure 2-4 illustrates the time-multiplexing display principle.

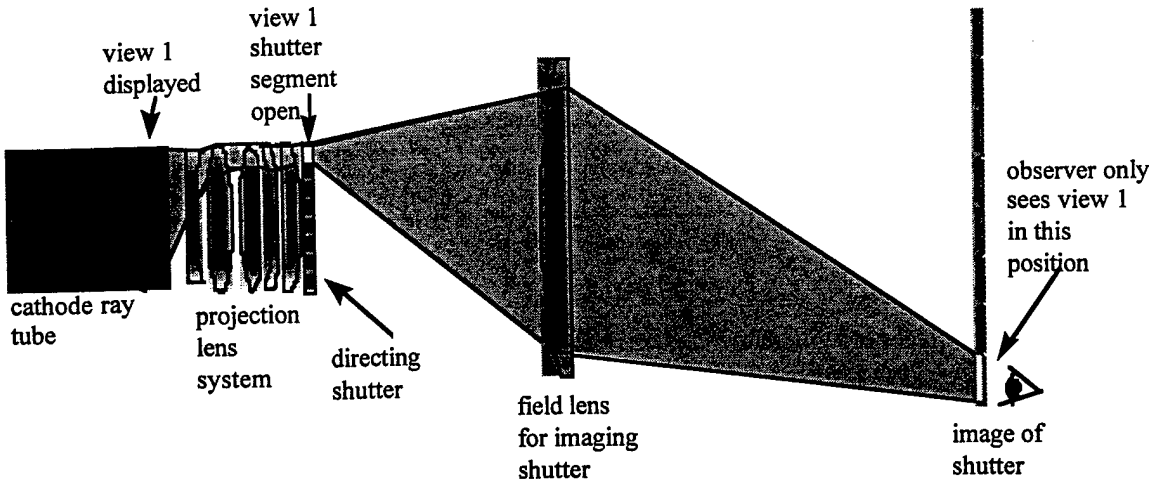

Figure 2-4: Litton Time-Multiplexed Autostereo Display

2.3.3 Demonstrations

The basic game network is illustrated in Figure 2-5. The network consists of PC platforms and either a 2D or 3D display, and connected by ethernet. The 3D display is driven by four graphics generators. The game play is actuated by a mouse on the 2D platform and by a trackball on the 3D platform.. RTI 1.0-2 (Windows NT) was installed on both platforms.

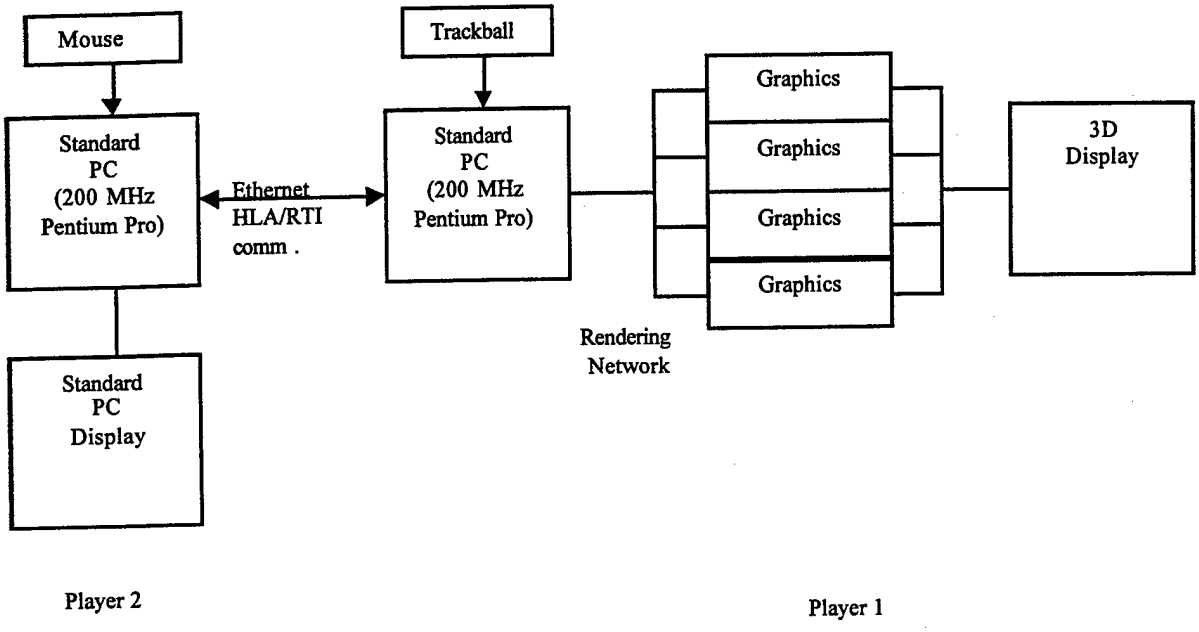

Figure 2-5: Networked 3D Autostereo and Standard Display Block Diagram

This configuration was used for FOM development and implementation of the realtime interactive 3D Tong game. Joint development at Litton and at Perceptronics was possible since the platforms were compatible. Demonstrations of the interactive game using both 2D and 3D displays were held at Litton, where the prototype 3D display is resident, and at the Interservice Industry Training Systems and Education Conference (IITSEC), held in Orlando, Florida, December, 1997. The working networked Tong game was demonstrated at IITSEC to STRICOM program management and technical monitor personnel.

2.4 Ownership Management and Transfer of Ownership

2.4.1 Problem Statement.

In a distributed 3D game, ownership of particular entities is highly important for game playability, particularly for fast-action games. Ownership refers to which playing entity is responsible for determining relevant characteristics of the moving entity, such as position, speed and orientation. Transfer of ownership is the point at which control of the entity passes from one platform to the other, and the optimal point is dependent partly on network performance.

Network latency can impact the way object positions are perceived as compared to their real position in the game environment. Consider the Tong ball as a game entity. A player may easily miss the ball if his perception of the ball trajectory deceives him with a wrong position. Indeed, the entire outcome of a fair competition may be negatively impacted, if certain players experience network delays and others do not. Transfer of ownership may be used in anticipation of a close interaction occurs in order to avoid network latency effects.

2.4.2 Implementation of Ownership Transfer in RTI 1.0.

The implementation of ownership transfer distinguishes between the following two types:

A. Transfer of ownership if the object state required for the simulation is completely known by the receiving federate.
B. Transfer of ownership if additional object state information is required for the simulation by the receiving federate.

We call the part of the object state that is known by the receiving federate prior to the ownership transfer the *public object state* and the part of the object state that is not known the *private object state*. In ownership transfers of type A, there is only a public object state. In ownership transfers of type B, the object state is comprised of both public and private parts.

If the object state is already known by the receiving federate (type A), the ownership transfer is implemented by the following sequence of HLA Ownership Management services:

1. The owning federate initiates the ownership transfer by invoking the NEGOTIATED form of the requestAttributeOwnershipDivestiture service asking to divest all attributes that form the object state.
2. This triggers an invocation of the requestAttributeOwnership Assumption service at the receiving federate (among others). The federate answers the callback by accepting all offered attributes (i.e., the object state).
3. Subsequently, the owning federate receives an invocation of the attributeOwnershipDivestitureNotification service indicating that it should stop simulating the divested object.
4. Also, the receiving federate receives an invocation of the attribute OwnershipAcquisitionNotification service, indicating that it should start simulating the acquired object.

Ownership transfer of type A has been implemented in the Tong prototype as described above and proved to be a viable method to hand over the ball between the two players. There is no noticeable impact on the perceived game play during the ownership transfer process.

If an additional object state is required by the receiving federate (type B), the ownership transfer can be implemented by the following sequence of HLA Ownership Management services:

1. The owning federate initiates the ownership transfer by calling the NEGOTIATED form of the requestAttributeOwnershipDivestiture service asking to divest all attributes that form the public object state.
2. This triggers an invocation of the requestAttributeOwnership Assumption service at the receiving federate (among others). The receiving federate answers the callback by accepting all offered attributes (i.e., the public object state).
3. Additionally, the receiving federate determines the set of private attributes that it doesn't know about, but requires for the simulation. The federate eventually needs to

add the private attributes to its list of subscribed attributes and invoke the `subscribeObjectClassAttribute` service with that list, followed by an invocation of the `requestClassAttributeValueUpdate` service with the private attributes. Then the federate invokes the `requestAttributeOwnership Acquisition` service for the private attributes.

4. Subsequently, the owning federate receives an invocation of the `attributeOwnershipDivestitureNotification` service, indicating that it should stop simulating the divested object. The federate also receives an invocation of the `provideAttributeValueUpdate` service with the set of private attributes, which it answers by eventually changing its list of published attributes and invoking the `publishObjectClass` service with that list, followed by an invocation of the `updateAttributeValues` service with the private attributes. Then the federate receives an invocation of the `requestAttributeOwnershipRelease` service. The federate answers the callback by releasing all requested attributes. Afterwards, the federate may fall back to the original set of published attributes.

5. Also, the receiving federate receives an invocation of the `attribute OwnershipAcquisitionNotification` service with the set of public attributes.

6. Later, the federate receives an invocation of the `reflectFederateObject` service that provides the requested values for the private object state, followed by the `attributeOwnershipAcquisitionNotification` service invocation with the set of private attributes. Afterwards, the federate may fall back to the original set of subscribed attributes.

Ownership transfer of Type B has not been implemented yet in the Tong prototype. It is planned to design and implement a complete set of ownership transfer schemes (i.e., ownership initiated by owner/non-owner and with/without private attributes) during the Phase II using the modified ownership management services of RTI 1.3. Also, it appears to be beneficial to provide a higher-level support by splitting public attributes form private attributes in the FOM (i.e., those that are only required by processes that want to simulate objects). Further investigation into how this may best be provided within the HLA framework is necessary and is planned for future work.

2.4.3 Ownership Transfer and Data Distribution Management

Data distribution management (DDM) is a RTI service which allows data to be sent only to relevant participants, and therefore serves to reduce network load when exercises are scaled up to many participants. Data distribution management adds a level of complexity to ownership transfer. In the ownership transfers described above, knowledge about the public state was guaranteed by the constraint that the receiving federate, which invokes the requestAttributeOwnershipAssumption service, had to be subscribed to the set of public attributes and thus knows their latest values. In a DDM-style federation, knowledge about public state is further constrained by the subscription regions.

In general, two cases can be distinguished:

I. The transferring object falls within a subscription region of the receiving federate.
II. The transferring object does not fall within a subscription region of the receiving federate.

In case I, the transfer schemes of type A and B are directly applicable. In case II, the commitment that the receiving federate makes in step 2 of the transfer schemes A and B by specifying a return value in the `requestAttributeOwnershipAssumption` service, keeps the federate from subsequently requesting values for those attributes from the owning federate. The commitment already could have triggered the invocation of `attributeOwnershipDivestiture-Notification` at the owning federate, which then isn't able to provide those values anymore.

A potential solution would be to reject the assumption entirely, followed by the request of all the public (and private) values and then initiate a new acquisition process for the entire set of attributes by invoking the `requestAttributeOwnership-Acquisition` service. With the modified ownership management services of RTI 1.3 a different solution is possible. The commitment is not given in the `requestAttribute OwnershipAssumption` invocation, but by invoking the `attributeOwnership-Acquisition` service. This allows the receiving federate to delay the invocation until he has requested and received all the public (and private) values.

If case-II ownership transfer needs to be supported, further experience with and determining how it would best be provided within the HLA framework is necessary. An open question, for example, is how the receiving federate would determine the subscription region(s) that need to be used to receive all the public (and private) values for the transferring object.

2.5 Sharing Viewpoints and Display Independence

2.5.1 Problem Statement.

Sharing viewpoints is a feature that is widely used in entertainment (e.g., in movies) and also adds value to online game applications. Participants employing viewpoint sharing may learn about the game rules through observing the actions of players engaged in game play. Another application lies in providing exciting views through viewpoints at locations in the game with most interesting activities (e.g., racing cars competing for positions).

The Tong prototype provides viewpoint sharing to the observers. Observers can chose between their own viewpoint, which they can maneuver freely within the 3D world constraints, and a set of viewpoints that are either provided by other observers or by objects in the world (e.g., the players and the ball).

2.5.2 Implementation of Viewpoint Sharing in RTI 1.0.

Networked support for viewpoint sharing has been implemented by a FOM entry for the camera that provides both location and orientation.

```
(class Camera
    (attribute Location    FED_BEST_EFFORT FED_RECEIVE)
    (attribute Orientation FED_BEST_EFFORT FED_RECEIVE)
  )
```

Separation of the camera objects from the other game objects allows for keeping subscriptions to cameras independent from subscriptions to other game objects (i.e., the racquets and the ball). In fact, players do not subscribe to camera objects, only observers

do. Furthermore, the separation allows for exploiting MOM-style control in order to turn viewpoint sharing on or off. Viewpoint sharing is not essential for the game play, but it is a feature that would allow observers to participate in a game where observation itself is of value. An example is a race car game, in which being a fan in the stands is a worthwhile experience. Viewpoint sharing, however, causes additional network traffic. If, for example, network congestion demands it, priority support between players and observers suggests turning off viewpoint sharing entirely. The observers then can only follow the course of the game from their own viewpoints.

2.5.3 Extension for 3D Display.

The rendering interface that has been used in the Phase I prototype implementation does not require any special support in the camera object. An observer at the prototype 3D display is supported in the exact same fashion as an observer at a standard 2D display. The rendering software automatically generates the 28 cameras needed for the auto-stereoscopic display from the one camera object.

There are, though, certain aspects that may need to be added for a "3D camera" for optimal perception results, such as the separation distance between the 28 camera positions being used in the prototype autostereo display used in Phase I. When the autostereo display is used in the two- screen mode, in which viewers see different images when looking from the left of the screen and from the right, 14 camera positions are used for each image. This implementation may mandate a different 3D camera object. Likewise, such attributes may be useful for other displays in use now for games and other applications, such as head-mounted displays and head position trackers.. The representation of the humanoid in the VRML working group proposal, for example, provides a pair of eye positions, essentially supporting two cameras, separated by the eye distance.

2.5.4 Sharing Viewpoints with Objects Subject to Transfer of Ownership

When viewpoints are provided by objects, there is a causal relationship between the position and orientation of the object and the position and orientation of the camera that is modeled according to the object. If both the object and the camera are located in the same federate, the relationship may easily be established by the federate. However, if one of them is transferred to another federate, the causal relationship needs to be maintained. Otherwise it may result in artifacts, such as in the case where the camera view operates on a

previous position of the object and is mostly obstructed by the object whose next position is used for rendering.

2.6 Multiple Levels of Detail Viewing

2.6.1 Problem Statement

Whenever observers share their viewpoints with others, additional data flow between federates is necessary. Attributes such as eye position, viewing direction, and viewing angles must be updated by the original observer as they change so that all observers sharing one viewpoint see the same things. Complete multiplicity in spatial or temporal resolution requires significant network resources. This additional network traffic may compete for the limited network resources with the attributes that players need for a fair game play.

In order to avoid an impact on the outcome of the game, general mechanisms to prioritize between the two participating groups are needed. In situations where the network cannot accommodate all the requirements of all the participants, viewing at lower levels of detail becomes a necessary tradeoff. Observers may receive state data at a lower rate (spatial tradeoff) and may also synchronize less frequently with the view of other participants (temporal tradeoff). Data distribution management services in the HLA may be used to limit the data flow.

In a commercial application, players and other participants in the game are expected to use equipment with a wide range of performance characteristics. To allow for a fair and enjoyable game experience, those characteristics that might influence the outcome of a game need to be considered also.

2.6.2 Results

Results using DDM are not possible with RTI 1.0, since DDM is not implemented in RTI 1.0. We will continue research using RTI 1.3 when it is released.

2.7 Test Results and Performance Estimates

Each game application which uses network resources typically must make tradeoffs between network load and desired game performance to preserve the integrity, fair play, and playability of the game. In general, tradeoffs fall in two areas: (1) Tradeoffs of

network capability, platform implementation, and level of detail, so the game play is seamless and within player expectations regardless of detail present in the particular gaming area (e.g., number of active entities); and (2) Tradeoffs in acceptable perception by the players, so that the game retains its interest and integrity in all types of realtime game interaction.

2.7.1 Objective Network Performance.

Use of the RTI in conjunction with particular platform implementations allows analysis of performance as a function of network communication methods and platform idiosyncrasies. Isolation of performance decrements to their source will allow determination of applicability of the RTI to meet the requirements of present interactive games. Our goal is to develop methods to provide a good balance between interactivity and game details including priority options between players and observers

Our findings to date are encouraging for continued use of the RTI in interactive game applications. We have found that slowdown in game play is due to performance of the rendering software used in the prototype 3D display, and not to the RTI. This finding is leading our partners Litton and Infinity to search for other 3D rendering methods.

2.7.2 User Perception

Acceptance of an interactive game experience is ultimately determined by user evaluation, since performance tradeoffs are transparent to him. Our initial plan was to use the 3D display's split screen capability to show a user different test sequences with no delay between trials. We have instead used 2D displays to examine user perception issues.

Preliminary findings are that the user perceives a noticeable lag in motion, and a corresponding change in his ability to react, for remote versus local objects. Users experience the lag much more on the true 3D display, where the ball clearly seems to move away from them, as opposed to the 2D perspective display where they use other cues to track and play the ball. Our expectation is that findings in this area are determined largely by the specific characteristics of our Tong game, the implementation and the platform capability. In general, perceptual optimization of games appears game specific, and game developers will use unique optimization methods for each game.

2.8 Conclusions and Next Steps

We were able to demonstrate a playable realization of a networked, interactive 3D game based on HLA/RTI principles, and addressing several major implementation issues, including transfer of ownership and sharing viewpoints. In addition, we demonstrated display and platform independence, an important prerequisite for commercial game application where home platforms vary in configuration and capability. Our findings are encouraging for continued use of the RTI in interactive game applications. The Phase I effort thus served its intended purpose of validating the original research assumptions and clarifying the work necessary for continued development.

We identified a number of areas which will require further work to determine more overall application of HLA principles and the RTI for commercial applications. Some of these areas can be addressed when later versions of the RTI which implement other services are available. It is planned that these will be addressed as part of Phase II should it be awarded. These issues are:

1. *Type B Transfer of Ownership.* Type B Transfer of Ownership is necessary when object state information in addition to that known by the federate is required. We plan to implement Type B in conjunction with release of RTI 1.3,
2. *Multiple Levels of Detail Viewing.* Levels of detail greatly influence required data flow, which can be minimized by using the Data Distribution Management service of the RTI. Since Data Distribution Management is not implemented in RTI 1.0, we will address multiple levels of detail viewing when DDM is implemented in RTI 1.3.
3. *Game Performance and Display Interaction.* We found significant game play slowdown when the autostereo display was used. The slowdown in not due to the RTI but rather to rendering software. Litton and Infinity are presently searching for other rendering methods using internal funding. We will continue game performance measurement and optimization when the improved rendering method is installed.
4. *Testbed Upgrade.* To address scalability, LAN/WAN and heterogeneous hardware for game applications, we will continue work with an upgraded testbed with more platforms and integration of a large screen autostereo 3D display under development now at Litton. Since optimization appears to be game specific, we will also implement simple prototype games representing standard classifications of interactive games, including sports, role playing, and persistent environments with many participants.

Each of these will be performed as part of our larger effort to develop a comprehensive, open architecture for gaming based on HLA principles so the achievements and advantages of Department of Defense interactive simulation can be brought to the growing universe of online games.

APPENDIX: TONG FILE LISTING

./BANG.WAV
./DebugLocal
./DebugLocal/SCI10.dll
./DebugLocal/Tong.bsc
./DebugNet
./DebugNet/SCI10.dll
./include
./include/AttributeTranslatorTemplate.cpp
./include/AttributeTranslatorTemplate.h
./include/BallController.h
./include/Controller.h
./include/Debug.h
./include/DS.H
./include/EntityFactory.h
./include/GameViews.h
./include/Line.h
./include/Mathematics.h
./include/paddle.h
./include/PaddleController.h
./include/RadarDisplay.h
./include/resource.h
./include/resources.h
./include/RMFunction.h
./include/RtiTypeMap.h
./include/Set.h
./include/Vector.h
./include/Win32Controller.h
./include/win32view.hh
./include/xwshColors.h
./include/Display.h
./include/RealimationDisplay.h
./include/Realimation.h
./include/RtiTypeMap.cpp
./include/Set.cpp
./include/Model.h
./include/UserController.h
./include/CameraController.h
./include/Shape.h
./include/EntityManager.h
./include/JgrFederateAmbassador.h
./include/Camera.h
./include/View.h
./include/TongClasses.h
./include/Entity.h
./include/RtiInterface.h
./include/GameEngine.h
./Tong.dsw
./Tong.ico

./Makefile
./resource.h
./src
./src/Controllers
./src/Controllers/Controller.cpp
./src/Controllers/PaddleController.cpp
./src/Controllers/Win32Controller.cpp
./src/Controllers/BallController.cpp
./src/Controllers/UserController.cpp
./src/Controllers/CameraController.cpp
./src/Controllers/JgrFederateAmbassador.cpp
./src/Engine
./src/Engine/3dtypes.h
./src/Engine/config.h
./src/Engine/DEFS3D.H
./src/Engine/dhr.h
./src/Engine/GENERAL.H
./src/Engine/geom.cpp
./src/Engine/geom.def
./src/Engine/geom.h
./src/Engine/GLIB3D.H
./src/Engine/I_DEFS.H
./src/Engine/i_ext.h
./src/Engine/I_MACS.H
./src/Engine/I_TYPES.H
./src/Engine/mathlib.h
./src/Engine/memory.cpp
./src/Engine/memory.def
./src/Engine/ML_3D.DEF
./src/Engine/ML_DEFS.H
./src/Engine/ml_euler.cpp
./src/Engine/ml_euler.def
./src/Engine/ml_exts.h
./src/Engine/ml_fix.def
./src/Engine/ML_FLOAT.Cpp
./src/Engine/ML_FLOAT.DEF
./src/Engine/ml_int.def
./src/Engine/ml_macs.h
./src/Engine/ML_MATRX.DEF
./src/Engine/ML_NORMS.Cpp
./src/Engine/ML_NORMS.DEF
./src/Engine/ML_PROTO.H
./src/Engine/object.cpp
./src/Engine/object.h
./src/Engine/paddle.cod
./src/Engine/PIXEL.H
./src/Engine/PLATFORM.H
./src/Engine/SCI10.h
./src/Engine/scMessage.h
./src/Engine/serial_controller.h
./src/Engine/SL_GEN.Cpp
./src/Engine/SL_GEN.DEF
./src/Engine/VPOOL.Cpp
./src/Engine/vpool.def

./src/Engine/Paddle.cpp
./src/Engine/collision.cpp
./src/Entity
./src/Entity/Entity.cpp
./src/Entity/Camera.cpp
./src/Entity/EntityManager.cpp
./src/Game
./src/Game/GameViews.cpp
./src/Game/Tong.cpp
./src/Game/EntityFactory.cpp
./src/Game/GameEngine.cpp
./src/Makefile
./src/Model
./src/Model/Model.cpp
./src/Model/Shape.cpp
./src/Mouse
./src/Mouse/Debug
./src/Mouse/Debug/SCI10-d.lib
./src/Mouse/Debug/SCI10.dll
./src/Mouse/mouse.cpp
./src/Mouse/SCI10.cpp
./src/Mouse/SCI10.dsp
./src/Mouse/SCI10.dsw
./src/Mouse/SCI10.opt
./src/Mouse/SCI10.plg
./src/Mouse/serial_controller.cpp
./src/Mouse/serial_controller.h
./src/Mouse/SCI10.ncb
./src/Network
./src/Network/DATABASE.C
./src/Network/IMGHNDLR.C
./src/Network/INITIAL.C
./src/Network/MAIN.C
./src/Network/RealiNet.cpp
./src/Network/UPDATE.C
./src/Network/RealiLocal.cpp
./src/Network/KEY.C
./src/Sound
./src/Sound/DS.CPP
./src/Util
./src/Util/Line.cpp
./src/Views
./src/Views/RadarDisplay.cpp
./src/Views/RealimationDisplay.cpp
./src/Views/View.cpp
./src/Views/Win32View.cpp
./src/Views/RtiInterface.cpp
./Tong.fed
./Tong.ini
./Tong.rbs
./TongNet.ini
./Tong.rc
./Tong.plg
./Tong.ncb

./Release
./Tong.dsp
./Tong.opt
./Tong.list